An Introduction to Object-Oriented GIS in Archaeology

André P. Tschan
Institute of Archaeology. Oxford University.
36, Beamont Street, oxford, OX1 2PG. U. K.
E-mail: andre.tschan@arch.ox.ac.uk

Introduction

Geographic Information Systems (GIS) are increasingly applied to holistic, inter- and intrasite investigations.[1] This powerful, computer-based technology accommodates a great variety of environmental, ecological and cultural information, by means of selective data capture, management, manipulation, analysis, modelling and spatial referencing (Ruggles, 1992: 108). However, GIS programs, like any other tool, are subject to a specific set of application characteristics (*i.e.,* operational and functional parameters), that prescribe the process for integration, presentation and interpretation of the archaeological record. Based on the chosen software and hardware, there will be inherent conceptual and structural design limitations, that directly determine, or affect, the outcome of a study. Furthermore, success in obtaining acceptable and meaningful results can be severely restricted, depending on the quality of the data and, more important, the capacity of the GIS package to process the information, implemented in the form of modelled abstractions. This work looks at a new technology for archaeological investigations, by providing an introduction to Object-Orientation (OO)[2] and to Object-Oriented Geographic Information Systems (OOGIS), which represent the basis for an ongoing evaluation into the potential of this tool.

Object-Orientation in Archaeology

OOGIS is by no means the only domain to adopt an Object-Oriented platform and structure (Booch, 1994: 78); others include software packages for air traffic control, animation, databases, operating systems, telecommunications, *etc.* In fact, commercial OOGIS seem to be among the most recent tools to make use of an OO design.[3]

Consequently, with regard to the role of OOGIS in archaeology, there is a marked absence in the general literature. Nonetheless, some specific attempts to introduce this technology have been made, with mixed success, including an investigation into isostatic uplift in Finland, using an OOGIS program called Miljöflex (Nunez, et al., 1995).[4] More current applications, using purpose built programs for archaeology that incorporate an OO approach in spatial analysis and modelling include *FieldNote* by Dr. Nick Ryan at the Computing Laboratory, University of Kent at Canterbury[5] as well as *CRISys*, co-developed by César González Pérez, as part of the Grupo de Investigación en Arqueología del Paisaje of the Universidad de Santiago de Compostela.[6]

As a trend, there does seem to be an increasing interest, with regard to Object-Orientation within the discipline. In addition to the above, there are also a few publications available, that involve OO, but which are either concerned with application areas other than GIS, or provide a more general introduction into the potential of this technology. Examples of some of these include a look at archaeological database design (Stine and Lanter, 1990: 80-89, Feder, 1993: 221-227), abstract data structures in GIS (Ruggles, 1992: 107-112), the theoretical implications of raster, vector and OO systems (Zubrow, 1990: 69-71), Object-Oriented designs for excavation simulation (Barroca and Rahtz, 1992: 39-48), and OO for system analysis (Rold, 1993: 213-220).

A Need for Change?

OOGIS uses currently available microsystems hardware, while incorporating a comparable plethora of methods and routines, common in standard GIS packages (Smallworld, 1991: 11). However, one of the questions to address is whether the latest gadgets, or theoretical models, come as a spontaneous consequence of bad practice. In other words, do we really alter our behaviour, or approach, from one day to the next, because the current way is suddenly considered too limited, or outdated?[7] By just implementing and advocating the most fashionable, and perhaps not always the best suited, products, it is easy to engage in a dangerous process of high-tech, punctuated equilibrium and, as a result, to end up exhibiting a complete disregard for any real, evolutionary understanding of the quantitative and qualitative changes in the data and the generated results.[8] In essence, due to the novelty value and the claimed advances, older and sometimes, well-proven technologies can become targets for replacement, with "newer, faster and better" tools, without properly assessing the potential benefits and consequences.

At present, the most popular and technically versatile GIS packages are either raster or vector-based (Weibel 1997: 113).[9] Despite the fact that they often incorporate similar routines for processing spatial input, they are fundamentally different, with regard to their conceptual, structural,

[1] A fact exemplified by the number of works (328) compiled for a bibliography of GIS in archaeology (Petrie, et al 1995).
[2] OO = [Object-Orientation or Object-Oriented], depending on the grammatical context.
[3] Smallworld, one of the more successful OOGIS packages, was first introduced in 1990 (Smallworld 1991: 5).
[4] Although the authors point out that Miljöflex has great potential for Cultural Resource Management, it is deemed limited for archaeological research (Nunez, et al 1995: 147).
[5] http://www.cs.ukc.ac.uk/research/infosys/mobicomp/Fieldwork/papers/
[6] http://www-gtarpa.usc.es/Proyectos/crisys/index%20en.htm
[7] A tool should not automatically be considered useless just because it is not Object-Oriented (Feder 1993: 113).
[8] The long-standing debate on the advantages and disadvantages between raster and vector which persists to date (Weibel 1997: 113) should serve as an example of this dilemma.
[9] Nowadays, many GIS programs can also accommodate both raster and vector data.

presentational, and analytical capabilities. Within this context, OOGIS should act as an enhancement and not as a replacement for established technologies, because even as a new tool, it still shares those overarching criteria, which define a computer program as a Geographic Information System – incorporating data collection, storage and retrieval, processing, analysis, and reporting facilities (Peuquet and Marble, 1990: 10).

There is, however, the requirement for applications, using OOGIS, to identify and prepare data in such a way that it can fit the new OO model for past or present, natural, or cultural phenomena. The basic premise is that Object-Orientation allows us to employ improved concepts and tools, in our desire to create a close resemblance of a particular view of the "real" world (Henderson-Sellers, 1997: 13, Khoshafian and Abnous, 1995: 7).); for which, in return, we have to change the way we perceive and define our surroundings, when engaging in an Object-Oriented GIS study.

A (GIS) World View

Due to the fairly broad nature of the general theme (*i.e.*, GIS in archaeology), it is obvious that at first, there must be some distillation process, which takes into account design issues for the available types of GIS technology. Then, followed by a description of the fundamental concepts, relating to Object-Orientation,[10] the desired outcome should be a contextual understanding of both major components (GIS and OO), driven by archaeological theory and analysis.

Undoubtedly, it is crucial to thoroughly understand the detailed processing parameters, with respect to any selected GIS program, in order to develop an application successfully. But, at the same time, the quality in representing our world, or the world of a distant ancestry, using a GIS, will inadvertently relate to the tool's capacity for modelling complex sets of phenomenological abstractions. In other words, when introducing a new technology, there is often the promise of some improved, information handling. However, and as in the case of OOGIS, any such enhancements might not be identified, by the changes in the specific operational characteristics of the GIS, but rather come as the result of a new data model (the way a view of the world can be represented). The following sections, exemplified by potential archaeological scenarios, aim to describe the conceptual and overarching design characteristics for raster, vector, and Object-Oriented GIS, and how each manifests itself, in the form of the data model, data structure, data representation, and topology.

A GIS Application Framework

This work is not intended to establish a qualitative assessment, with regard to any GIS technology. There is a deliberate emphasis in the use of generic and descriptive explanations, in order to avoid establishing a hypothetical, or even controversial, ranking system.[11] It seems clear that each technology incorporates a set of useful application characteristics; otherwise, they would not find general and continued usage in many disciplines, including within archaeology. The primary purpose is to introduce OOGIS, through highlighting the respective differences between raster, vector, and OO, with regard to the main conceptual domains.

Overall, any study or research, regardless of the selected GIS technology, will require some prior and explicit thought, about the following design and implementation aspects (Table 1):

Table 1: General GIS Application Domains	
Data Model	*The selected view of the world and its contents*
Data Structure	*The composition of the data within the GIS*
Data Representation	*The visual display parameters for the data*
Topology	*The relationship properties of the data*

When looking at the proposed schema in detail, there might be an issue, with regard to the semantics and overall definitions used. For example, the choice of "data model" and "data structure" can be blurred at times, the likely result of a colloquial misuse, and the popularity of the latter idiom (Bartelme 1995: 35). Nonetheless, it is essential to define and adhere to strict distinctions, when trying to identify any differences between standard GIS technologies and OOGIS, particularly, when a major change involves an overarching concept, rather than just a series of fancy, functional computer keystrokes.

There is also no detailed separation of the compositional elements (storage, management, *etc.*) that make up the data structure domain, although, this does not deny the fact that major aspects, of internal configuration, warrant an operational understanding, before engaging in a study. But, when opting for an affordable, commercial GIS package, one already tends to focus on relevant, structural components. As a result, the desired program characteristics and any explicit preference for a database type (*i.e.*, relational or OO, using a local or remote filestore, *etc.*) will be satisfied, by acquiring the appropriate product. Furthermore, when equipped with good linking facilities, the specific, inherent data structure may be of limited importance, to those application developers, using other software tools to generate and maintain their data collections, thus limiting the GIS functionality, purely to processing, or analysis.

Once again, the aim is to highlight major, conceptual differences for each technology (raster, vector, and OO) based on the proposed application domains, and not to describe the in-depth, internal characteristics of individual and currently available GIS packages.[12]

[10] Explaining OO and OOGIS using only a few pages will be incomplete at times since it normally requires a whole book to address all the detailed aspects relating to this technology and research tool.

[11] The basic premise is that the application needs should determine the choice of tool based on the requirements for the desired results (Feder 1993: 223).

[12] This description of the data model, data structure, data representation and topology only addresses the fundamental differences because any extensive detail would have been beyond the scope and topic of this paper.

The Data Model

A general definition of a GIS data model might be: "a human conceptualization of the world (*i.e.*, a selected view), which identifies the parameters and entities[13] relevant for an application." In essence, it is the description of any real, or abstract, objects, using a varying degree of complexity and accuracy, in the process of identification (Bartelme, 1995: 17). These data model definitions may be augmented further, to include: "a measurement framework (*i.e.*, time, space, and attributes) combined with a representational scheme" (Chrisman, 1997: 23). Worboys (1995: 23), on the other hand, suggests an even more detailed data model construct comprised of the first four stages in a GIS application life cycle (Table 2), an ordered development process, defined by its (1) *phenomena*, (2) *abstraction*, (3) *conceptual computer model* (without an actual GIS tool in mind), and (4) *computational design phase* (incorporating specific data structure criteria, like relational database, OO, *etc.*) – where all four stages are prior to the actual (physical), GIS implementation.

Table 2: A GIS Life Cycle: Data Model	
1. Application Domain	*Real-world phenomena (natural and cultural)*
2. Application Domain Model	*Abstraction*
3. Conceptual Computational Model	*Design without data structure criteria*
4. Logical Computational Model	*Design with data structure criteria included*

Regardless of which definition one prefers, the basic premise remains, that the data model acts as the required interpreter, between phenomenological information and the GIS. In other words, the *source domain* (real world) is translated by the data model, via abstraction and simplification, into the *target domain* (*i.e.*, the assigned GIS package), in order to become available for further processing within this latter context (Worboys, 1995: 145).[14] This, then, also implies that any analysis, performed by a GIS routine, will in return, be available for the interpretation of the actual source entities, relationships, and the like.

A Raster Data Model

The basic assumption for a raster GIS, data model suggests that an entire study area can be divided into smaller sections, subject to their homogeneous thematic content (Bartelme, 1995: 46). This process, also known as a *field-based* approach, uses *tessellation* to transform a selected surface into a regular framework of abstract spatial distributions which is formalized as a mathematical construct, like grid cells, or pixels (Chrisman, 1997: 65). In essence, any collection of real-world phenomena, as identified by the researcher, gets a field assignment, when draping a grid array over the study area. The final outcome is a data model, which represents the world as a regular layout, composed of individual cells, and where each cell contains attribute values, according to the specific location, or entity, that might occupy its space.

This technical description may be best highlighted by a hypothetical, archaeological example (Figure 1: Raster GIS), where SITE A is a single representative of a *thematic layer*[15], called Iron Age Sites. What is important to know is that, regardless of the actual world size, the modelled measurements for SITE A will always be subject to the dimensional definition of the grid cells (Weibel, 1997: 113).[16] Hence, when using a raster GIS, the issue of resolution (the size of the grid cells, for the array to be placed over the world) determines the overall quality of the abstraction. Therefore, prior to the actual implementation, the data model becomes the most important source of analysis, utilized to determine the precision, or detail, as part of the specified requirements for a study.

A Vector Data Model

A vector GIS generally employs an *object-based*, data model, although the choice can be between exclusive usage, or a combined model, that additionally includes raster information.[17] What this means is that, real-world entities are identified by their spatial characteristics, in order to be represented as part of an "exact" computer model. The actual process of abstraction, concerning natural and cultural phenomena, involves geometric primitives (*i.e.*, point, line and area)[18] subject to a specific location in the form of coordinate values, within a geo-reference system (Chrisman, 1997: 62). Hence, the modelled entities primarily incorporate meaningful and possibly complex information, through their visual definition. But, because point, node, and line constructs tend to initially describe the drawing structure attributes, they don't actually support OO and all-inclusive, "object" definitions, which can contain a whole host of contextual excavation and analysis data (Zubrow, 1990: 70).[19]

The vector data model is also exemplified in Figure 1 (Vector GIS). In this case, SITE A has been modelled, by using a point as its geometric entity representation. Some of its associated, drawing attribute values are shared with other entities, and together they form a thematic *coverage*[20],

[13] The use of "entities" for real-world spatial and cultural objects is an attempt to avoid confusion with "objects" as defined in the Object-Oriented sense, which describes a computer construct.

[14] It is clear that since the two most popular (spatial) data models to date use raster and vector technologies, Object-Orientation must yet establish itself as a viable third alternative.

[15] Several thematic layers overlaid on each other represent a composite map in a raster GIS (Bartelme 1995: 46).

[16] Whether or not SITE A in reality is only 10m x 12m is ignored because the proposed example defines 50m x 50m cells and "A" will therefore be modelled by filling the extent of the cell with a uniform value.

[17] "Object-based" should not be confused with "Object-Oriented" since vector GIS is primarily concerned with the representation and accurate geometric coding of entities. OOGIS, on the other hand, builds complete entity (object) models that simply incorporate all characteristics as attributes.

[18] Where area can also include and define surface representations (*e.g.*, digital terrain (DTM) or elevation (DEM) models).

[19] The situation may be remedied by the fact that database tables can be linked to the vector GIS entities as part of the data set. This results in a decentralized and distributed data organization which still does not represent a true "object" model but clearly aids to improve the overall information content.

[20] Although often used synonymously with "layer", a "coverage" as a digital overlay of attributed spatial data (Langram 1993: 50) is a term that pertains more to vector GIS due to the varying degree of areal cover potentially

depicting an Iron Age site distribution. However, because of its quality of precision, a vector GIS excels in accurate, spatial definitions and has its own set of rules, that affect this data model. For example, the actual boundaries for "A", when identified as part of a real-world phenomenology, could have easily been represented using the exact extents. Hence, the alternative over the point representation, for SITE A, was to draw a polygon, that demarcates the appropriate expanse.

This decision, on how "A" is represented, is based on the detail, defined by the data model and, therefore, directly affects the amount of additional information, which needs to be stored in database tables and linked to the entity. Suffice to say, that the cartographic appearance, of a vector GIS, does allow a very precise abstraction of the real world, by virtue of its representational quality. However, to fully benefit from this data model, a clear and thorough application development (conceptual and physical) is demanded, which defines what, how, and where archaeological information is integrated into the GIS.[21]

An OO Data Model

OOGIS uses an Object-Oriented design, but can make exclusive, or inclusive, use of either of the aforementioned data models. However, unlike raster or vector, the OO world view is an intuitive construct, inherently linked with human perception and the understanding of our surroundings. *Objects* represent real-world entities and their properties, while an OOGIS acts as a mediator, between the world and the way we perceive and model it (Freska and Barkowsky, 1996: 110).[22]

This means that any real entity (*e.g.,* a "ditch") or abstract concept (*e.g.,* "feature" for all non-portable cultural remains), situated within a given environment, is defined by the specific compositional characteristics and treated as a distinct inhabitant of this contextual setting. Hence, with an OO approach, natural and cultural phenomena are modelled as complete, *object* definitions, using *object classes* within the OOGIS. In other words, these (geo-referenced) objects can incorporate all the *attributes* and *behaviour*, present in the real-world entities, which not only uniquely, distinguishes them from each other, but also establishes existing *relationships* between them, either as a result of spatial (proximal location)[23] or typological associations.

Within an OOGIS, at first glance, there does not seem to be a clear distinction, using the archaeological example provided (Figure 1: Object-Oriented GIS), inasmuch as SITE A looks identical to the vector image. This is basically an issue that pertains to the data representation, as subject to computer graphics, and it is described further, below. The difference, actually, relates to how this particular, Iron Age site has been identified as a tangible phenomena, and its subsequent conversion to an abstract object.

In essence, with the OO data model, one first looks at all the components that inhabit, or define, a particular space. Then, the major constituent parts are selected, to form the overarching object classes, using the relevant attributes and other compositional characteristics (behaviour). Within this context, SITE A represents a single *instance* of the object class, "Sites", and the specific and identifiable characteristics deemed important for "A", as a real-world entity, are incorporated in this class definition.[24] This includes structural, as well as representational aspects (*i.e.,* the *drawing geometry*), in order to achieve the closest compositional resemblance to the original and observable phenomenon. Objects are formed by the constituent parts, or traits, of an entity, and there is no artificial separation in the OOGIS, or by the OO data model. Hence, OO allows an homogeneous representation of an archaeological reality, as encountered in the field, which is modelled by maintaining the integrity of the world as we perceive it.

The Data Structure

The data structure transforms the data model into the computational model. A world view is integrated in a GIS, using an arrangement of entities which permit the construction of relationships, through software operations (Chrisman, 1997: 57). The data structure, therefore, refers to more technical aspects than the data model, and, clearly, the most crucial element is the systematic organization of the selected space (Nievergelt and Widmayer 1997 : 186), which, inadvertently, pertains to the structural composition and storage of spatial data, for either a raster, vector, or Object-Oriented GIS.

With regard to raster, the basic data structure is a Cartesian grid array. This (possibly geo-referenced) matrix defines a regular construct of cells, containing values (*i.e.,* height, colour, *etc.*) divided into rows and columns. The example provided in Figure 1 (Raster GIS), shows that each cell represents a real-world surface area of 50m x 50m, and when using a possible archaeological case, we can see that SITE A has a specific location, occupying an entire cell, which also includes its associated attribute information:

Row = *5*
Column = *12*
Height = *250m*
Colour = *9 (red in the specified colour scheme).*

However, every pixel, as the smallest indicator of any data variation, has to be stored as one record, and even though many might be identical, the image file storage will contain a direct model of the total number of mapped cells (Martin, 1996: 108-109).[25] This means that all cells combine, to form

leaving open spaces between entities. In Arc/Info terminology, a layer is conceptual whereas a coverage is physical (Hadzilacos 1996: 242).

[21] This is a decision process that can easily result in an unproductive overemphasis for either the visual representation (*i.e.,* too much detail in the visual representation which in turn contains too few database references for analysis) or the distributed data storage (*i.e.,* an extensive database table construct linked to a drawing composed of a graphical distinction between entities that is too limited).

[22] An object in the real world is a unified construct which incorporates and controls all its distinguishing characteristics by virtue of its definitional name.

[23] An interesting quote by Waldo Tobler (Worboys 1995: 145): " ...the first law of geography: everything is related to everything else, but near things are more related than distant things."

[24] An object class is the template for each instance (object).

[25] It may be worthwhile to point out that an image file for raster is not a database in the common sense. It also would be foolish, to say the least, to attempt to enter each pixel in a real database because of the excessive storage demand (Bartelme 1995: 279). Although, compression methods can

single, thematic layers, in this case, "Iron Age Sites", as part of a continuous surface data structure.

Vector GIS, on the other hand, only contains values for those objects that actually occupy some information space, defined by a coordinate system (local, regional, UTM, or some other geo-referenced architecture). Furthermore, a distinction needs to be made between the spatial and non-spatial attribute data; the latter tends to be held in separate databases, while control is maintained through an entity's ID value (Martin 1996: 97). This is exemplified by Arc/Info, which uses such a hybrid architecture. Hence, the Arc system files (or filestore) control information on the spatial and graphical structure (*i.e.*, point, line and area geometries, colour, text, *etc.*, as well as topology), while Info (as a commercial DBMS) holds all other associated data (Worboys, 1995: 285).[26] For SITE A, in Figure 1 (Vector GIS), this means that the unique identifier (ID = "A") is associated with an entity's location (geo-reference) while at the same time, describing the characteristics of this geometric drawing entity, as part of a thematic coverage:

Northing = *4271775*
Easting = *476575*
Altitude = *250m*
Geometry = *Point*
Colour = *Red (a selection from a colour palette)*
Text = *A (based on a font and size)*
Coverage = *Iron Age Sites*

This information forms the basis of the spatial data, whereas all additional, non-spatial attributes (*i.e.*, excavation dates, finds, artefact types, *etc.*, associated with SITE A) would be contained within an external DBMS and in the case of a relational database, through a (possibly) complex system of interrelated tables.

Finally, OOGIS employs a fully integrated, object-data structure, that can also adopt the two, aforementioned technologies, either exclusively or in combination. However, unlike raster or vector, OO identifies each (geo-referenced) phenomenon, as belonging to one of a range of object classes, which means it is especially well suited for the natural and cultural data, that can be easily conceptualized as discrete objects (Martin, 1996: 105-106). This also results in an integration of spatial and graphical, as well as non-spatial attributes, for objects, and further facilitates the construction of seamless, spatial databases (Worboys, 1995: 287-288).[27]

Any processing can, therefore, be performed on an object directly, which is in stark contrast to the plethora of external database tables, that may be affected by the same operation, when using a vector GIS. Furthermore, behaviour is included and applies to each instance of an object class. This means that programming routines, called *methods*, described further in the OO concepts section, below, can perform dynamic processing procedures[28], which also represent the most crucial part in the development of the data structure for an OOGIS, because objects can only be accessed, via their pre-defined methods (Taylor, 1990: 135).[29]

Using the archaeological example in Figure 1 (Object-Oriented GIS) we can see that SITE A is an instance of the object class, "Sites", which, in turn, prescribes the attributes and behaviour routines, as part of the conceptual abstraction of a real-world phenomena (*i.e.*, all archaeological sites, regardless of cultural and chronological distinctions). This results in an all-inclusive description for SITE A:

Object Class = *Sites*
SITE A
ID (Instance) = *System-defined*[30]
Northing = *4271775*
Easting = *476575*
Altitude = *250m*
Period = *Iron Age*
Geometry = *Point*
Colour = *Red (defined for object class)*
Text = *A (font and size for object class)*
calculate_distance_to_freshwater = *800m (create: calculate value using this object class (behaviour) method)*

It needs to be pointed out that the behaviour method, "calculate_distance_to_freshwater", further explained in section 5.4, could be dynamic (update), inasmuch as changes in the path of the river, once recorded, trigger a renewed calculation, and hence, the current distance value is perpetually established, with respect to SITE A's proximity to this freshwater source.

Data Representation

The data representation has already been touched upon, through the explanations of the data model and the data structure. Hence, raster, which has close ties to the physical layout of computer graphics hardware, divides a geographical region and its contents into an uninterrupted surface of uniform fields, using a grid cell or pixel array (Chrisman, 1997: 65-66). This way, all cells have (attribute) values associated with them, as part of a continuous image display. Vector drawings, on the other hand, use a series of geometric primitives (*i.e.*, point, line, and area polygons) to describe entities within a specific location reference scheme; in addition, raster data can also be integrated, possibly as a background map, in a vector GIS (Clark, 1992: 16). Similarly, OOGIS can apply a combination of raster and vector, as part of its visual construct and interface.

Thus, the specific requirements of a study should determine the data representation quality. In other words, the desired world view dictates whether raster is sufficient, or a more

alleviate the problem of huge image data files by grouping continuous and contiguous cells.

[26] In practice it is possible to link other relational databases to Arc and there is no proprietary restriction.

[27] A spatial database is seamless when devoid of any artificial boundaries that could be encountered by a user.

[28] Performing any variation of the basic *create* (independent/dependent), *destroy* (permanent) and *update* (transformation) operations (Worboys 1995: 175).

[29] Where all methods will be defined as part of an object.

[30] Each object has an internal identity independent of any attribute value which means that it is unnecessary to provide an explicit instance variable for the purpose of identification (Worboys 1995: 86-87). It is therefore possible to distinguish between two separate objects even when they have the same instance values.

complex vector depiction is needed, or, possibly a combination of both. At the same time, it does not imply that a raster GIS is the least desirable, and that vector, or Object-Oriented, GIS packages have better representational facilities. It is merely an issue of tool selection, depending on the topic of research and required extent of detail.

Based on the examples in Figure 1, it would be easy to say that the first picture (Raster GIS), were it to be actually generated, using a raster GIS program, is quite sufficient for the purpose of demonstrating the provided (hypothetical) archaeological scenario. As a result, the two subsequent images (Vector GIS and Object-Oriented GIS) would be identified as "over the top", for actual illustration purposes. In the case of SITE A, we, therefore, are presented with a choice of whether or not to use a basic raster square cell, in red, to be indicative of "Iron Age Sites", or to alternatively use a geometric point representation, and all its necessary associated (internal/external) attribute storage constructs by applying either a standard vector GIS or an OOGIS package.[31] Decisions along these lines are fundamental in the design of a GIS application, and data representation plays a crucial role in selecting the appropriate tool for the job.

Topology

In everyday life, we often tend to take spatial relationships (*i.e.*, proximity, left/right, inside/outside, connected/disconnected, *etc.*) for granted. In contrast, this information actually forms a key element in a GIS (Clark, 1992: 20). Referred to as the *topology* of a system, this data needs to be specifically declared as entered values. Topological information is generally used with vector data structures, which make clear distinctions between entities (*i.e.*, point, line, and area representations), since it is much harder to encode explicit, spatial relationships using raster technology (Martin, 1996: 105, Chrisman, 1997: 64).[32] In essence, a connected network of nodes is required, and on its most basic level, topology defines the operational distinctions for *boundary* and *interior* spaces, which can be further expanded, to include *overlap* and *covering* (Worboys, 1995: 173).[33]

Because spatial relationships are integrated as attribute values, within a GIS, the same issues, as pointed out in the data structure section, are applicable. In other words, and excluding raster, topology is stored externally in system files, or a database, for vector GIS, while OOGIS includes this information as part of its object class definition. In Figure 1 (Vector GIS and Object-Oriented GIS), we can imagine that the river has some relationship with SITE A: possibly identifying the distance from freshwater, in answer to some research question. However, the vector program would need to determine the closest node, as an explicit value, selected from the chain of nodes, created to represent the river. On the other hand, any topological information using an OO approach simply defines connected instances of object classes (*e.g.*, objects of type "Sites", and, for the purpose of this example, objects of an additional class called, "Rivers", for all rivers in the study area), which automatically establishes the closest node reference as part of this relationship. The distance from SITE A to a river can then be calculated at run-time through a behaviour routine (method) incorporated in the relevant and (topologically) associated, object class definitions.

Object-Orientation

The chances are that archaeologists, in many cases, will have gained exposure to other OO programs, prior to applying an OOGIS package (current programming languages (Visual Basic and Object Pascal) or contemporary databases, are likely examples). However, for all OO tools, the main concepts and data model criteria remain the same, while the application design changes, according to the task for which the program is intended. Whether or not it is a GIS for investigating the past, or a database product to record excavation discoveries, is circumstantial. What is important to know is how this technology affects the data, as well as its ability to present our world, and that of our ancestors, using a specific model for phenomenological abstraction. For this purpose, a general development of Object-Orientation and the differences, between standard programming techniques and OO, are explained further.

Procedures, Modules and Objects

Object-Orientation, as a conceptual idea, may certainly be older than the tangible 30 years, since the inception of Simula67,[34] developed by the Norwegians, Ole-Johan Dahl and Kristen Nygaard (Henderson-Sellers, 1997: 1, Khoshafian and Abnous, 1995: 13). Yet, another 25 years passed, before the current and large mass appeal for OO technologies was identified as one of those "revolutionary", paradigm shifts for computer applications (Henderson-Sellers 1997: 6). The fact that this progression is by no means as dramatic, nor as swift, a punctuated event, as potentially indicated, by the above terminology, might be demonstrated successfully, by identifying the appropriate "evolutionary" milestones for this technology (Riel, 1996: 1). The overarching focus lies primarily with the conceptual ideas, that have been implemented, throughout the last 30 years, and which have lead up to Object-Orientation. The purpose is to identify some gradual, structural changes, while including the potential landmark events, or the arrival stages, of some particular machinery, only when deemed as contributory or essential information.[35]

[31] It should be emphasized that a point representation in a vector system is a drawing entity with associated information whereas in an OOGIS the geometric style for the visual appearance of instances is merely an attribute pre-defined in object classes.

[32] A basic reason for raster being less able to incorporate topology lies in the fact that spatial relationships are present in each thematic layer and it takes all themes, once overlaid, to combine this fragmented information (Bartelme 1995: 150).

[33] There are numerous other spatial relationships that are defined within the relevant literature (*e.g.*, ref.) depending on specific entity representations as being connected or disjoint; explained in more detail therein and beyond the scope of this work.

[34] A general purpose programming language, although mostly used in simulation modelling.

[35] The emphasis is to avoid the standard exercise of listing ad nauseam the so-called "key moments" or "influential" people important in the historical development of Object-Orientation when the chances are that this information might provide little or no contextual understanding.

Procedural Language Structures

For many of us, the most memorable progression in computer systems has to pertain to the basic user interface. Only 10 years ago a manual command-line (possibly, menu-driven), keyboard interaction was the standard. Nowadays, the most prominent form of software operation and management most certainly depends on graphical environments, or tools, and this characteristic is mostly due to products, wanting to compete successfully on a global market (Riel, 1996: 2). However, in the overall course of advancement this was a very recent step, and some prior developments, affecting Object-Orientation, warrant further description.

Without actually going all the way to the beginnings of computer design and programming languages, it is obvious that some backtracking is required, to, at least, the point where the by-all-means-still-thriving, procedural programming methodology makes its mark. This long-time standard and persistent approach, to digital data handling, advocates structured programming techniques. It is very interesting to note that during the early 1970's, there were simultaneous developments, inasmuch as Niklaus Wirth introduced the programming language, "PASCAL", which greatly initiated the procedural approach, whilst a group around Alan Kay, at the University of Utah, set down the basis for "Smalltalk", the first Object-Oriented coding tool (Beekman, 1994: 192, Henderson-Sellers, 1997: 1). Thus, these two, successful programming and application techniques exhibit an evolutionary pattern, that quite clearly suggests a parallel genesis from a common (unstructured) ancestry, and OO, therefore, does not follow a direct succession, or replacement path, from the current, procedural software architecture.

In general, the development of structured programming has been a great step in the right direction and still serves its purpose, concerning complex data manipulation and management. It would, therefore, be quite wrong to identify them as "incorrect" techniques, requiring urgent change. But they must be considered limiting, when it comes down to the current requirements, needed for handling ever-increasing and diverse information sets.[36]

Modules versus Objects

Another critical factor, in understanding the progressive developments, instrumental in what nowadays distinguishes OO from standard, coded procedures, pertains to the introduction of modularity. As a means to avoid the unsightly and often difficult to analyse "GO-TO" statements,[37] sub-programs, called modules, improve the general logic and cohesiveness of programs and applications, using a procedural design framework. Yet, modularity can not automatically remedy those cases, where poor design will also lead to bad code, and once the damage has been done, it is hard to return and fix the inherent, "drawing board" problems.

These sub-programs can also be wrongly compared with Object-Oriented objects, inasmuch as they both seem to be small, task-specific modules, and represent elements of a greater application, or program, structure. But, it is this point, where it is critical to understand that modularity differs considerably from OO, which imposes a completely new perspective on any software design. Most crucially, an object contains both data and instructions (Hadzilacos, 1996: 243).[38] Alternatively, the module is a collection of code lines, performing singular, or sets, of functions, as part of the overall program execution. This means that a user will be able to "do it", or "accomplish a job", rather than the highly flexible, Object-Oriented task handling environment, that allows decisions on "what can", or "what needs to be done" (Beekman, 1994: 195). This fundamental, conceptual difference represents not just a semantic contrast, but manifests itself quite dominantly in the life cycle of a program (from analysis to maintenance), as well as in the necessary computer code.[39]

Concepts of Object-Orientation

One problem, all approaches (procedural, modular, and OO) share, is the fact that programming languages have initiated the processes of change, before appropriate overarching analysis and design methods were developed (project management and modelling, being the last additions to a theoretical, OO application framework) (Henderson-Sellers, 1997: 9). There is, therefore, a need to look more closely at what concepts are involved in an OO implementation, with respect to archaeological circumstances, when applying this particular view of the world.

The most fundamental question, relating to the idea of Object-Orientation is the following: what comprises the definition of an object? The simplest explanation, in conjunction with archaeology, can be put forth by using a (non-spatial) analogy, involving pottery. Archaeological ceramics illustrate the case for an OO approach, quite clearly, by the particularly healthy understanding of their physical, functional, and symbolic characteristics. Although, we must not forget that the aim is to model the world, using more "natural" interpretations and definitions (Riel, 1996:2).

---

[36] One of the major problems affecting the procedural approach is that it tends to be a one-way ticket in a single direction. Consequently, each advancing development stage represents a departure from the point where overall design or analysis errors can still be corrected. This is a dilemma within an industry-, business- and also academia-oriented environment where there is an increasing need to accommodate a demanding deadline schedule.

[37] These riddled the particularly large unstructured programming code in pre-1970's third-generation languages like BASIC or FORTRAN. The programming language "generations" are:

First	=	machine language,
Second	=	assembly language,
Third	=	high-level languages,
Fourth	=	non-procedural, English-like languages,

where the basic progression is along the lines of each successive stage becoming easier to use and more like natural language (Beekman 1994: 195).

[38] The term *object* can describe either (1) a real-world phenomenon (entity), (2) a conceptual or mathematical abstraction (a triangle or a mountain), or (3) its exact technical definition in the OO program model. In addition, an object without behaviour is unfeasible since it can not exchange messages with other objects; therefore any object requires at least one programming routine (method).

[39] OO is a way to handle a subject (*e.g.*, a model for archaeological data) while OO as a technology applies to all steps in a system life cycle including analysis, design, implementation and use.

Object-orientation, therefore, forces us to adopt a new way of thinking, since it cannot be viewed as a mere extension of procedural, or action-oriented, methods.

Class and Objects

Each class, in essence, is a template that describes the state or structure of its objects (Khoshafian and Abnous, 1995: 3). If we, then, consider an arbitrary and deliberately, unspecified set of five pots, and aim for an initial and fundamental description, we would likely classify them broadly, as an assemblage of pots, or, better yet, simply as "Pottery" (an object class). Regardless of any further analysis, this point already exemplifies one of the fundamental, structural features of an Object-Oriented[40] approach, namely the ability to group a specific phenomenon, whether it is a real and tangible occurrence or an abstract notion.

Having established this object class, we can further elaborate, without actually looking at the pots themselves. The degree of in-depth knowledge, one has acquired, is quite irrelevant, when it comes down to describing some additional aspects of these pots. Without additional information, we might first apply a basic system, where each pot is uniquely identified, using a sequential code (*i.e.,* Object-1P, Object-2P, *etc.*), to represent individual instances of the previously created, "Pottery" class.[41] Within an archaeological context, certain cultural distinctions, among the assemblage, may also be assumed, in the form of separate spatial, regional, or temporal properties, as part of the imaginary data set, in order to apply a simple location, reference structure, where the pots might have been found.[42] Finally, and again, without actually needing any further inspection, we can assume decorative, dimensional, functional, and symbolic aspects for these artefacts, since they most likely will exhibit these traits (attributes), to a varying degree of complexity.

The importance of this example lies within our ability to conceptualise, or abstract, a whole series of factual information, without actually having seen or touched the artefacts themselves. Our *a priori* understanding, of the instances for "Pottery", is represented schematically (Figure 2) and could result in the following compositional structure, for the first of the five pots (Table 3):

Table 3: An A Priori OO Artefact Model

Object Class *Pottery*	
Attributes	**Instance Values**
ID (Instance)	*Object-1P*
Location	*Object-1F*
Colour	*Red*
Height	*26cm*
Width	*30cm*
Geometry	*Star*
Behaviour	
Type (determine)	*Jar*

The "geometry" and "type" variables need to be explained further, as part of this hypothetical archaeological example. The former has nothing to do with any actual phenomenological trait which might describe the shape or appearance of the five pots. It is a user-defined, attribute value that establishes the visual representation, or drawing properties, for the instances of "Pottery" in the OOGIS. Whatever a pot may actually look like, be it a real-world entity, on the screen or on a printout, it will be represented as a star-shaped symbol, indicating that the geometry attribute is an integral part of an all-inclusive, object definition.

The method "type", on the other hand, is a dynamic variable, and in this example it can be either subject to the interpretational expertise of individual researchers, a visual recognition software, or some programming code to generate more information for each of the five pots. It is obvious that the object class, "Pottery", is just an overarching, abstract construct, and that a further separation, into more refined object classes, is possible, based on the specific properties that characterise different types of archaeological ceramics. What is important is the fact that any object definition includes behavioural faculties, in order to communicate with other objects, when using its methods to send and receive data, via *messages.*[43] Hence, "type", when activated, is a means (behaviour) to determine typological distinctions for each instance of "Pottery", which in turn, produces an additional attribute value (*i.e.,* "*Jar*" for *Object-1P*).

Overall, this example represents the mere beginnings, in what could be a useful and logical OO artefact, data model, similar to the one that has application in ceramic analysis, already (Sinopoli, 1991: 52-53), particularly, with regards to a type-variety typology, which advocates an organisation, ranging from broad pottery classes, to detailed differentiation of ceramics, based on distinct diagnostic traits. It is, therefore, clear that an OO approach would and does lend itself to archaeological investigations, which focus on typological identification of artefact remains.

However, this work mainly looks at the particular implications of Object-Oriented GIS, with respect to spatial analysis in archaeology. A far greater level of abstraction is required, when trying to quantify an often, large-scale regional phenomenology (natural and cultural). Hence, unlike the basic example of "Ditchbury" (Figure 3), any real-world entities are likely to have much less, discernible

[40] The definition "Object-Oriented" is somewhat incorrect and the more appropriate description of "Class-Oriented" would be more accurate (Henderson-Sellers 1997: 27).

[41] This unique identifier would of course normally be generated by the system and would not need an explicit (user-defined) attribute value for each object within an OOGIS application.

[42] For this example, a sequential system based on the instance ID's of a new object class called "Features" is used (*i.e.,* Object-1F, Object-2F, *etc.*); which is introduced later as a case for OO abstraction of spatial phenomena in archaeology.

[43] Only by sending a message to an object can a desired method pre-defined for that object be invoked.

properties, due to such problems as indeterminate boundaries. Considering a description similar to the ceramics example, an *a priori* OO spatial model for the first feature (*Object-1F*), might therefore, look something like the following (Table 4):

Table 4: An A Priori OO Spatial Model

Object Class *Features*	
Attributes	**Instance Values**
ID (Instance)	*Object-1F*
Location	*Ditchbury*
Length	*25m*
Width	*10m*
Height	*1m*
Geometry	*Area*
Behaviour	
Type (determine)	*Ditch*

If we look at the example of the five archaeological pots, as the instances of an object class, called "Pottery", and the "Ditchbury" objects, belonging to the "Features" class, we can establish an essential, conceptual OO criteria (Figure 4: Class and Objects). The specific reason for the above examples, using a very basic structure of classes and objects, is to highlight well-known elements of archaeological material culture or spatial phenomena and to subject them to an OO data model. Despite not having any greater insight into the physical characteristics, provenance, or any other archaeological context, the OO view of the world assumes that a variety of attribute and behaviour patterns must be implicitly present, by using identifiers, like "Pottery" and "Features". The idea is that the actual name of a class implies its attribute and behaviour (Riel 1996: 12-13).[44] In other words, in OO, the class defines a phenomenon, through abstraction, while each object instance uses this class definition as a mould, to describe its properties.[45] A ceramic artefact is, therefore *instantiated* from the "Pottery" class, and the "Pottery" class, in turn, is the abstract generalisation of all the ceramics, that exist in the real world.

Polymorphism

Polymorphism, as one of the powerful concepts in OO, describes the use of methods (behaviour) that have the same name, in several different object classes (Taylor, 1990: 48, Worboys, 1995: 89).[46] Therefore, identical messages can be sent to objects, which may exhibit close, or even no, definitional similarities (Henderson-Sellers, 1997: 186-187). However, it is also apparent that as an integral part of any object class definition, specific methods need to be created first, in order to receive, or reply to external communications.[47]

Expanding on the above archaeological scenario, a general object class, "System", with a method, "type", is introduced, which can activate the behaviour routines for objects, including instances of "Pottery" and "Features" (Figure 4: Polymorphism), based on the circumstance that, through an *inheritance relationship* (described below), every object class will have a method, called "type". This allows for a simultaneous execution of "type", despite "Pottery" and "Features" objects, requiring different processing values to perform their typological evaluation routine, in this case, using a series of specific, attribute information as the source data.[48]

With regard to instances of the "Pottery" class, "colour", "height" and "width" provide the parameters for the automated analysis, performed by "type", which establishes a more refined distinction between ceramic artefacts (*e.g.*, jars, bowls, *etc.*), and returns the value of "*Jar*" for "*Object-1P*" (Table 5A). The same goes for 'Features", where the dimensional values "length", "width" and "height" are used as the required input for its "type" method, distinguishing, in more detail, different spatial phenomena (*i.e.*, ditches, pits, *etc.*), and which result in "*Ditch*" for "*Object-1F*" (Table 5B). It is, therefore, clear that through polymorphism, a vast number of methods can be defined as part of any object, in order to create and customise a highly flexible OOGIS application, according to a specific research design and avoiding the potential for generating fairly generic, "off the shelf" type of GIS study results, which may come as a consequence of rigid and pre-defined, operational design, framework limitations.

Table 5A:
The "type" Method Analysis for the "Pottery" Class

Object Class *Pottery*		
Attributes	**Instance Values**	**Type Analysis**
ID (Instance)	*Object-1P*	
Location	*Object-1F*	
Colour	*Red*	*Red*
Height	*26cm*	*26cm*
Width	*30cm*	*30cm*
Geometry	*Star*	
Behaviour		=
Type (determine)		*Jar*

[44] That this requirement is fulfilled is especially clear in the case of our imaginary ceramic assemblage where any imaginable decorative dimensional and functional characteristic that may apply can be identified and defined due to the specific and "natural" name for the object class, "Pottery".

[45] The class is the overarching template for objects inasmuch as it represents the basic abstract data type:

Pottery = Generalization of all ceramics,
Features = Generalization of all features,
in the examples provided.

[46] Methods can have the same name to perform more or less the same task but with a different outcome (*e.g.*, a "print" method may produce text output for one object while printing images for another).

[47] In addition, special overarching object classes comprised of only one instance can be defined which contain behaviour routines to globally interact with multiple objects (*e.g.*, "System" as a "root" object in an OOGIS). This is in line with the fact that all methods must have an explicit object class definition including the ones which are created as part of a general application domain.

[48] It is obvious that the attributes specified would never satisfy the input requirements for a fully operational automated typology analysis system. This example, therefore, serves only to illustrate an Object-Oriented concept (polymorphism) using a hypothetical archaeological scenario.

Table 5B: The "type" Method Analysis for the "Features" Class		
Object Class *Features*		
Attributes	**Instance Values**	**Type Analysis**
ID (Instance)	*Object-1F*	
Location	*Ditchbury*	
Length	*25m*	*25m*
Width	*10m*	*10m*
Height	*1m*	*1m*
Geometry	*Area*	
Behaviour		=
Type (determine)		*Ditch*

Inheritance

On the most basic level, inheritance refers to a *class hierarchy* system, defined by the degree of abstraction and detail in object class definitions. In other words, the *superclasses* (parents) are more generic, or abstract, than the *subclasses* (children) (Henderson-Sellers, 1997: 21). This also reflects a general, human understanding and knowledge structure, which employs concepts of *generalisation*, leading to *specialisation*.

Inheritance is also a mechanism, which allows sub-classes to represent special cases of a superclass and hence, they automatically inherit all characteristics from their overarching classes (Taylor 1990: 22),[49] while defining their own attributes and behaviour.[50] Much like biological taxonomies, when applying an inheritance, hierarchy system, in an OOGIS, a holistic and richer, semantic relationship, among entities in a given space, can be established (Khoshafian and Abnous, 1995: 82).

Broadening the two, aforementioned, "Pottery" and "Features" object class definitions, a series of sub-classes, for specific types of archaeological ceramics and features, are introduced (Figure 5). In this example, "Jar" and "Bowl" represent children of "Pottery", the same way as "Pit" and "Ditch" are descendants of the parent, "Features". Each of these sub-classes has been selected, to represent their own object class, but they also acquire or inherit all traits from the overarching superclasses based on a Shlaer-Mellor inheritance model (Starr, 1996: 94-95). This, then, means that each real-world jar or bowl is an example of "Pottery", and, similarly, any actual pits or ditches represent archaeological "Features". In turn, within an OOGIS, instances of "Pottery" are sub-classed and must be either a "Jar" or a "Bowl", and likewise, any "Features" will be identified as either a "Pit" or a "Ditch".

In general, all structural attributes and behavioural methods are inherited by the sub-classes, through their *superclass relationship*.[51] Looking again at our example of ceramic artefacts, it is, therefore, obvious, that a system-defined ID for an object, once established, is a value that remains constant, throughout the class hierarchy. However, the drawing geometry is not the same in "Pottery" and its children, because the superclass would likely use a general representation (*e.g.*, a point) for all types of ceramics, and the two subclasses, "Jar" and "Bowl", override the parent with their respective and specific display symbols (*i.e.*, a star and a circle), in order to visually distinguish between them, within the OOGIS application.

Furthermore, the method "type" for "Jar" and "Bowl" is inherited from "Pottery", and for each of these subclasses, this behaviour routine is distinct, as a result of requiring separate processing values for analysis. Therefore, "type" in "Pottery" represents the generic, automated process of evaluating typological definitions for ceramic artefacts.[52] This means that a message can be sent to the "type" method, for the instance "*Object-1P*" of "Jar", which in turn, executes the method "type" in "Pottery", using *Object-1P* as the target object. To explain this more simply, we need to clarify that methods are operations, designed to retrieve or update the state of an object, where the state of an object represents the stored attribute values (Khoshafian and Abnous, 1995: 104). For our example, the "type" methods, in either "Jar" or "Bowl", use a set of arguments, which merely add to, or modify, as required, the inherited behaviour routine, originally defined in "Pottery". This way, the appropriate variables from each sub-class are used in the analysis, to develop further typological distinctions (*i.e.*, subdividing "Jar" into "*Small Storage*", "*Liquid Storage*", "*Large Storage*", *etc.*, and "Bowl" into "*Cooking*", "*Serving*", *etc.*) (Table 6A).

With regard to the "Features" object class, a very similar hierarchy exists to the one described for archaeological ceramics. Again, the ID's for spatial objects are established by the system, maintained by the object, and recognized throughout the inheritance structure. The drawing geometry for the superclass also uses a generic representation (*e.g.*, a point), for all features, which is then overridden, by the display attributes of the subclasses, "Pit" and "Ditch" (*i.e.*, circle and area) (Table 6B).

[49] Also known as *structural* (attributes) and *behavioural* (methods) inheritance (Khoshafian and Abnous 1995: 82).

[50] Which potentially override (supersede) any characteristics bequeathed from the respective superclasses.

[51] It should be emphasized that, there is no actual instance (object) for the superclass but only one for the subclass, which contains all its own attributes and methods in addition to all the ones not overridden from the overarching superclasses.

[52] Whereas, "type" in "Features" represents a different behaviour routine specific to spatial phenomena; see below.

Table 6A: The Inheritance Class Hierarchy for the "Jar" Instance "Object-1P"		
Object Class *Jar*		
Attributes	**Instance Values**	**Available through Inheritance**
ID (Instance)	*Object-1P*	
Handles	*No*	
Spout	*No*	
Portable	*Yes*	
Geometry	*Star*	
		Superclass *Pottery*
		ID (overridden)
		Location
		Colour
		Height
		Width
		Geometry (overridden)
Behaviour		
Type (determine)	*Small Storage*	
		Superclass *Pottery*
		Type (determine)

Table 6A: The Inheritance Class Hierarchy for the "Ditch" Instance "Object-1F"		
Object Class *Ditch*		
Attributes	**Instance Values**	**Available through Inheritance**
ID (Instance)	*Object-1F*	
Linear	*No*	
Fence	*Yes*	
Geometry	*Area*	
		Superclass *Features*
		ID (overridden)
		Location
		Length
		Width
		Height
		Geometry (overridden)
Behaviour		
Volume	$250m^3$	
Type (determine)	*Defence*	
		Superclass *Features*
		Type (determine)

Similarly, the method "type", in "Features", represents the overarching, automated routine to evaluate typological distinctions for its sub-classes, within the hierarchy, by using the necessary attribute values from each sub-class, in order to subdivide further instances of "Pit" into "*Posthole*", "*Refuse*", "*Burial*", *etc.* and "Ditch" into "*Defence*", "*Field* System", "Enclosure", *etc.*, However, for our example, the "Ditch" subclass adds some additional behaviour, in the form of a method, called "volume". This measurement routine is unique to the sub-class, and it highlights the issue of generalisation and specialisation. For example, while the attributes and methods in "Features" are mutually shared by the "Pit" and "Ditch" sub-classes (*i.e.*, generalised from the two spatial phenomena), they have characteristics exclusive to their definition as an object class (*i.e.*, specialised traits, which are unique to each). Inheritance is a powerful tool in OO, where a complex hierarchy of object classes can be established, using a logical and "natural" composition, which makes intuitive sense and this is also a construct, close to archaeological typology systems, and hence, should come quite readily to this discipline.

Relationships

In an OOGIS, relationships (including topology) are directly established between objects.[53] In other words, a relationship is an abstraction of real-world associations[54] in the same way as an object is an abstraction of a real-world entity (Starr, 1996: 49). Hence, universal, conceptual relations need to be established, that are valid for all situations where objects interact with each other (Freska and Barkowsky, 1996: 112). At the same time, it is clear that there are different types of relationships[55], which are described by a variety of terms, but essentially relate to the same concepts (Henderson-Sellers, 1997: 28, Starr, 1996: 56) (Table 7):

Table 7: Relationship Types	
Henderson-Sellers	**Starr**
1. *Aggregation*	*Binary*
2. *Association*	*Associative*
3. *Inheritance*	*Supertype (Superclass)*

A point that needs to be made is the fact that relationships can also have their own attributes, independent of the involved objects (Worboys, 1995: 71). The process of relationship abstraction formalizes how objects interact with each other (Starr, 1996: 49). *Aggregation* refers to a basic connection between objects (*i.e.*, "is_part_of") and may be the closest to topology within an OOGIS. *Association* is the relationship type, which requires the services of one or more objects, to create, destroy, or update instances within another object class, based on their associative interaction (*i.e.*, a client/supplier affiliation).[56] *Inheritance* refers to the hierarchical structure (explained in the section above) for class relations (*i.e.*, "kind_of", or "is_like"), which allows reuse of attributes and methods, defined in the respective superclasses.

For the purpose of demonstrating an archaeological example, we are mainly concerned with aggregated and hierarchical

[53] A quote by D. Ingalls suggests that: "Instead of a bit-grinding processor-raping and plundering data structure, we have a universe [in OO] of well-behaved objects that courteously ask each other to carry out their various desires.", (Booch 1994: 97).

[54] Which holds systematically between instances of object classes in an OOGIS (Starr 1996: 47).

[55] Identifying among other variations of dependency, generalization and specialization "many_to_many", "many_to_one" and "one_to_one" relationships for objects.

[56] VAT charges on goods are a prime example where an associative relationship can be constructed inasmuch as the correct percentage calculation can be handled by a separate object which returns the value to the original transaction object in order to update and print the overall total to be paid on the receipt for the customer.

relationship types.[57] Figure 5 (Inheritance and Relationships) shows that in addition to the link, that exists for the two superclasses and their respective sub-classes, there is also a direct connection between "Pottery" and "Features". The latter, aggregative relationship has been established, in order to describe a positional reference for individual, ceramic artefacts. In other words, *Object-1P* has a "location" attribute value (*i.e.,* the place where it was found), which is in relation to a relevant instance (*Object-1F*), in "Features". Described as a many_to_one relationship, called "located_in", each "Features" instance can contain a spatial reference for many "Pottery" instances, but a single ceramic artefact can only have one "location" value.

At the same time, the sub-classes, "Jar", "Bowl", "Pit", and "Ditch" interact with their respective superclasses using a many_to_one relationship, called "kind_of". This also means that actual archaeological jars or bowls are single "Pottery" instances, but this object class can describe many of these real-world, ceramic artefacts. The same applies to "Features", where numerous pits or ditches can be recorded as instances of this object class, but where each spatial feature can only have a single reference to its superclass.

In general, when establishing relationships for an OOGIS application, one danger is that the focus is too specific, on objects, when it should be on the connections, that exist between them. This undermines the overall quality and success of a study, since the power of a system depends largely on the amount and types of relationships implemented (Starr, 1996: 50), and requires a careful, and more substantial, analysis than the one provided in this example, which serves purely as a simplified archaeological demonstration.

Encapsulation

Another important concept in Object-Orientation is the idea of *encapsulation*, which combines data and methods and hides them from view - in line with a natural extension of the *information-hiding* strategy, developed in structured programming (Taylor, 1990: 31). While abstraction aims to define some visible behaviour for an object, encapsulation tries to hide the controlling mechanisms from general visibility (Booch, 1994: 49). Basically, the emphasis lies in a deliberate attempt to hide the operating details from any potential users, while the same system processes remain visible to other objects (Henderson-Sellers, 1997: 16).

The philosophy is one, that aims to only provide access to necessary information, and which lets any functional and implementation aspects of an object remain *private*, while the behaviour methods, with which objects communicate, are available through the *public* interface (Figure 6: Encapsulation).[58] In other words, the data can only be accessed by the object's methods, performing the standard tasks of reporting, storing, and calculating values (Taylor, 1990: 31). This process is managed by messages, which advise a receiving object, to carry out an indicated method, and to return the result of that action. In contrast, relational database structures use a *call-by-value* approach, where entities are connected to other entities by their values. OO data, on the other hand, is addressed indirectly, because of encapsulation (Worboys, 1995: 87), thus, protecting the information from corruption by other objects.

Figure 6 demonstrates information-hiding, for instances of "Pottery" and "Features", which can be accessed by the "System" method, "type", defined earlier in polymorphism, and the relationship, "located_in", that exists between the two object classes. It is clear, though, that a whole host of additional behaviours is needed, should we want to make *Object-1P* and *Object-1F* functional objects in a real OOGIS application.[59] However, this example serves its purpose, inasmuch as it shows how information is encapsulated, and what means of access are available.

The actual values for *Object-1P* and *Object-1F* are private and hidden. This is highlighted by their containment within the inner ellipse for each object diagram. Access to this information is gained by their methods, as defined in their respective object classes, and represented by the outer, object ring in the same diagram. In other words, a message can be sent to either individual, or multiple, instances of "Pottery" or "Features", by means of the "System" method, "type", in an attempt to communicate with the respective behaviour of each object. This also shows that a plethora of application, object classes can be integrated in an OOGIS, which supplement overall object communication between objects, through specialised methods.[60]

With regard to the aggregative relationship for "Pottery" and "Features", it has already been established that the "location" attribute value is a positional reference for ceramic objects, which are linked, via "located_in", to their respective spatial objects. This indicates that relationship definitions act much like methods, although they are far more specific in generating the appropriate means for interaction, between instances of object classes. In an OOGIS application, encapsulation plays an important role in the preservation of the overall, compositional integrity of real-world entities, which includes how they interact and correspond with each other, thus, permitting access only through a series of different behaviour methods, protecting the actual object data.

Advantages and Disadvantages for OOGIS

Based on the topical analysis to date, the final section of this work briefly highlights some of the already identified characteristics of OOGIS, in a summarised fashion. This list of advantages (Table 8A) and disadvantages (Table 8B), for application of this GIS technology in archaeology, should indicate the general potential, before some actual case studies can substantiate, or identify, additional benefits/limitations. At the same time, this summary should not be viewed as a qualitative assessment of OOGIS, in relation to raster or vector systems, but rather as a compilation of issues, that

[57] It is worth mentioning that for this example, messages can be sent bi-directional indicating that all objects can communicate with each other.

[58] All data should be private and public methods need to be specifically designed to access desired pieces of information; although private methods can be implemented indicating that not all methods must be public.

[59] Including the desired input and output *accessor* methods.

[60] Which introduces an additional level of flexibility to access encapsulated data.

suggest that this is a possibly, useful tool for archaeological research.

Table 8A: Potential Advantages
1. *There is a growing interest in OO technologies, within archaeology, as witnessed by recent work.*
2. *OOGIS should be considered an enhancement, and not a replacement, for raster or vector technologies.*
3. *Mapping is done, according to a continuous and seamless design, which allows the integration of spatial and cultural data objects, that normally span more than one traditional map sheet.*
4. *Objects represent abstractions of real-world entities and their properties; hence, an OOGIS acts as a skillful mediator, between the way we perceive the world and the way we model it.*
5. *An object ideally describes an abstract representation of a real-world phenomenon, using a specific, semantic definition (i.e., descriptive name).*
6. *The data structure is all-inclusive, by combining attributes and behaviour, into a holistic construct.*
7. *OOGIS can use raster and vector representation methods, and the graphical characteristics are part of each object class definition.*
8. *Topology is integrated as part of object class definitions.*
9. *There is a clear, conceptual and organisational class hierarchy system for objects, which manages the relationships between individual object classes.*
10. *Within the overall OOGIS design, object methods (behaviour) are incorporated, to allow interaction and communication between objects.*
11. *OO uses encapsulation, which ensures data security, by only allowing access through object methods.*
12. *OOGIS is an active research tool, and it should not be considered a one-off data store and display utility.*

Table 8A: Potential Disadvantages
1. *Objects need to be clearly defined spatially, a distinct problem in archaeology, considering researchers' discretionary practices (i.e., those "grey" or "fuzzy" areas, regarding concepts like site dimensions or general functional interpretations).*
2. *It may be more difficult to integrate new object classes, within the main (parent) control levels of an existing OOGIS class hierarchy, thus, demanding a well-developed application design and implementation strategy, at the beginning of any project.*
3. *The necessary hardware/software costs involved, in order to run an OOGIS operation, may still be higher than for a standard GIS implementation, using raster or vector technologies.*
4. *The training in OOGIS may require a considerable time for novices in GIS, as well as for people skilled in raster and vector application design since the new data model requires a different conceptualisation process, for natural and cultural phenomena.*

Conclusion

In general, a framework for archaeological fieldwork has been developed, which accepts change in practices, over time, in line with methodological improvements. However, with regard to OOGIS, there is not even an agreement, yet, on a conceptual application strategy for archaeological investigations.[61] In other words, there will certainly be a long line of differing, "expert opinions", before any acceptance of OOGIS, as an effective tool among the palette of GIS products used within the discipline, is achieved. This work must, therefore, be seen as one of the steps that introduces OOGIS to archaeology, while actual case studies will have to act as later reference material, to develop a concise and useful implementation framework. Consequently, the descriptions of hypothetical archaeological cases, throughout this introduction to OOGIS, and as basic as they may be, should still provide some insight into the application potential of this tool. Overall, more research will have to follow, particularly, since OOGIS may hold the key to modelling spatio-temporal relationships, which would then come a step closer to a fully holistic GIS, analysis methodology.

Acknowledgements

I would like to thank Dr. Gary Lock for his continued support and assessment of this work and Patrick Daly for his expertise on the operational details of vector GIS programs. Additional thanks goes to Francesco Menotti, for his clear review, and Sarah Semple, for her much appreciated input on archaeological artefacts and features. Furthermore, Jan Pieters of EDS Netherlands BV, as well as Dr. John Atiayh and Derek Hunter, of Smallworld UK, deserve explicit mention for their invaluable know-how on Object-Orientation and OOGIS. Dr. Nick Ryan and César González Pérez are mentioned here as a result of personal discussions, which confirmed the potential, and defined some interesting areas for future research, using OOGIS. Last, but not least, I greatly thank my wife, Colleen, for whom the world will never look the same, now that objects, which seem to be everywhere, have invaded our lives (and they need dusting!).

References

BARROCA, L., RAHTZ, S. (1992), *Object-Oriented design for excavation simulation programming*, in: LOCK, G.R. & MOFFETT J. (eds.), *Computer Applications and Quantitative Methods in Archaeology 1991*, BAR International Series S577, Tempus Reparatum, Oxford.

BARTELME, N. (1995), *Geoinformatik - Modelle, Strukturen, Funktionen*, Springer-Verlag, Berlin.

BEEKMAN, G. (1994), *Computer Currents - Navigating Tomorrow's Technology*, The Benjamin/Cummings Publishing Co. Inc, Redwood City.

BOOCH, G. (1994), *Object-Oriented Analysis and Design - With Applications*, Addison-Wesley Publishing Company, Menlo Park, Second Edition.

[61] Future research will therefore need to engage in some analysis that specifies the general application requirements in addition to the development of a theoretical implementation strategy for archaeological investigations.

CHRISMAN, N. (1997), *Exploring Geographic Information Systems,* John Wiley and Sons Inc, New York.

CLARK, M. J. (1992), *The GIS Survival Guide,* University of Southampton. GeoData Institute.

FEDER, J. (1993), "MuseumsIndex - An object oriented approach to the design and implementation of a data driven Data Base Management System", in: ANDRESEN, J., MADSEN, T., SCOLLAR, I. (eds.), *Computing the Past: Computer Applications and Quantitative Methods in Archaeology 1992,* Aarhus University Press, Aarhus.

FRESKA, C., BARKOWSKY, T. (1996), "On the Relations between Spatial Concepts and Geographic Objects", in: BURROUGH, P.A., FRANK, A. U., (eds.), *Geographic Objects with Indeterminate Boundaries - GISDATA 2,* Taylor & Francis, London.

HADZILACOS, T. (1996), "On Layer-based Systems for Undetermined Boundaries", in: BURROUGH, P.A., FRANK, A. U., (eds.), *Geographic Objects with Indeterminate Boundaries - GISDATA 2,* Taylor & Francis, London.

HENDERSON-SELLERS, B. (1997), *A Book of Object-Oriented Knowledge - An Introduction to Object-Oriented Software Engineering,* Prentice Hall PTR, New Jersey.

KHOSHAFIAN, S., ABNOUS, R. (1995). *Object Orientation - Second Edition,* John Wiley and Sons Inc, New York.

LANGRAM, G. (1993), *Time in Geographic Information Systems,* Taylor & Francis, London.

MARTIN, D. (1996), *Geographic Information Systems - Second Edition - Socioeconomic applications,* Routledge, London.

NIEVERGELT, J., WIDMAYER, P. (1997), "Spatial Data Structures: Concepts and Design Choices", in: VAN KREVELD, M., NIEVERGELT, J., ROOS, T., WIDMAYER, P., (eds.), *Algorithmic Foundations of Geographic Information Systems,* Springer-Verlag, Berlin.

NUÑEZ, M., VIKKULA, A., KIRKINEN, T. (1995), Perceiving Time and Space in an Isostatically Rising Region, in: LOCK, G.R., STANCIC, Z., (eds.), *Archaeology and Geographic Information Systems: A European Perspective,* Taylor & Francis, London.

PETRIE, L., JOHNSON, I., CULLEN, B., KVAMME, K. (1995), *GIS in Archaeology - An Annotated Bibliography,* Sydney University Archaeological Methods Series 3. Sydney: Archaeology (P&H). University of Sydney, Sydney.

PEUQUET, D.J., MARBLE, D.F. (1990), "Geographic information systems: an overview", in: PEUQUET, D. J., MARBLE, D. F., (eds.), *Introductory Readings in Geographic Information Systems,* Taylor and Francis, London.

RIEL, A.J. (1996), *Object-Oriented Design Heuristics,* Addison-Wesley Publishing Company Inc, Reading Massachusetts.

ROLD, L. (1993), "Synthesis in object oriented analysis" in: ANDRESEN, J., MADSEN, T., SCOLLAR, I. (eds.), *Computing the Past: Computer Applications and Quantitative Methods in Archaeology 1992,* Aarhus University Press, Aarhus.

RUGGLES, C.L.N. (1992), "Abstract Data Structures for GIS Applications in Archaeology", in: LOCK, G.R. & J. MOFFETT (eds.), *Computer Applications and Quantitative Methods in Archaeology 1991,* BAR International Series S577, Tempus Reparatum, Oxford.

SINOPOLI, C.M. (1991), *Approaches to Archaeological Ceramics,* Plenum Press, New York.

SMALLWORLD. (1991), *A Corporate Statement,* Smallworld Systems Limited, Cambridge.

STARR, L. (1996), *How to Build Shlaer-Mellor Object Models,* Prentice Hall PTR, New Jersey.

STINE, R.S., LANTER, D.P. (1990), "Considerations for Archaeology Database Design", in: ALLEN, K.M.S, GREEN, S.W., ZUBROW, E.B.W., (eds.), *Interpreting Space: GIS and Archaeology,* Taylor & Francis, London.

TAYLOR, D. A. (1990), *Object-Oriented Technology: A Manager's Guide,* Addison-Wesley Publishing Company Inc, Reading Massachusetts.

WEIBEL, R. (1997), "Generalization of Spatial Data: Principles and Selected Algorithms", in: VAN KREVELD, M., NIEVERGELT, J., ROOS, T., WIDMAYER, P., (eds.), *Algorithmic Foundations of Geographic Information Systems,* Springer-Verlag, Berlin.

WORBOYS, M. F. (1995), *GIS - A Computing Perspective,* Taylor & Francis, London.

ZUBROW, E.B.W. (1990), "Contemplating Space: A Commentary on Theory", in: ALLEN, K.M.S, GREEN, S.W., ZUBROW, E.B.W. (eds.), *Interpreting Space: GIS and Archaeology,* Taylor & Francis, London.

All Figures in CD-ROM.